A Grid Portal for an Undergraduate Parallel Programming Course

Juan Touriño, *Member, IEEE*, María J. Martín, Jacobo Tarrío, and Manuel Arenaz

***Abstract*—This paper describes an experience of designing and implementing a portal to support transparent remote access to supercomputing facilities to students enrolled in an undergraduate parallel programming course. As these facilities are heterogeneous, are located at different sites, and belong to different institutions, grid computing technologies have been used to overcome these issues. The result is a grid portal based on a modular and easily extensible software architecture that provides a uniform and user-friendly interface for students to work on their programming laboratory assignments.**

***Index Terms*—Grid computing, grid portal, parallel programming, supercomputing.**

I. Introduction

SINCE 1998, the Department of Electronics and Systems (DES) at the University of A Coruña, A Coruña, Galicia, Spain, has offered a one-semester elective course in parallel programming for undergraduate fifth-year computer science (CS) students. The inclusion of parallel processing in an undergraduate degree has been widely justified since the 1980s [1] since parallel computers have become much easier to use and much more widely available. It continued to be justified during the 1990s [2]–[4], and nowadays many universities offer specific courses to provide CS undergraduates with real parallel-processing experience [5] as an essential component to teaching parallel-programming concepts effectively. Three additional reasons have favored the introduction of a parallel-programming course in the CS curriculum at the University of A Coruña.

1) There is an active research group on parallel processing at DES.
2) There are powerful supercomputers available at the Supercomputing Center of Galicia (CESGA) [6] in Santiago de Compostela (70 km away from DES), mainly used for scientific computing.
3) There is growing interest of small and medium enterprises of Galicia in cluster architectures [7] because they deliver outstanding parallel performance at a competitive cost.

Manuscript received July 2, 2004; revised November 30, 2004. This work was supported by a Teaching Innovation Grant (UDC-TIC03-057) from the University of A Coruña, A Coruña, Spain; by the Galician Government (Xunta de Galicia, Projects PGIDIT02TIC00103CT and PGIDIT04TIC105004PR); and by the CrossGrid European Project (IST-2001-32243).

The authors are with the Computer Architecture Group, Department of Electronics and Systems, University of A Coruña, 15071 A Coruña, Spain (e-mail: juan@udc.es).

Digital Object Identifier 10.1109/TE.2004.842888

CS graduates skilled in parallel-processing techniques are, therefore, requested by both academic/scientific and enterprise environments.

Problems that arise in teaching a course in parallel programming include the difficulty in having access to supercomputing resources and the time it takes to teach their effective use, as stated in [8]. Wilkinson and Allen discourage the use of resources of supercomputing centers [9] because the technicalities of supercomputers may be too much for an undergraduate course. Moreover, their access must be controlled carefully by the center, and perhaps the centers are not equipped to handle a large number of inexperienced students. Therefore, the choice is the use of a dedicated local network of workstations for parallel programming.

Since the course was intended to be highly practical, a hybrid approach was followed: use of local resources (both from DES and the School of Computer Science (SCS) at the University of A Coruña), mainly for training purposes, and access to remote supercomputers (available at CESGA) to assess the performance of parallel algorithms. This approach has several advantages: students are more motivated if they have access to real supercomputers, and they have the chance to implement the same algorithm using different architectures and parallel-programming models, which enable students to make objective performance comparisons (execution times, speed-ups, complexity of algorithm development, etc.). Moreover, their curiosity for new machines stimulates self-learning to go deeply into specific parallel-processing issues of the target supercomputers.

The computers used in the course are administered by different institutions: DES, SCS, and CESGA. As a minor drawback, tedious bureaucratic procedures are necessary to obtain accounts for the students (mainly, access to the supercomputing center). However, the major drawback is the impact on students in using a number of unrelated and geographically distributed systems. Thus, each student has several accounts on machines with their own local file system (which involves source code files being continuously transferred among the target machines), and with different access, use, and security policies (e.g., restricted Internet protocol (IP) access that makes it difficult to log in to the machines from home). Once the student is logged on, he or she has to deal with different compilers, libraries to be linked, and specific program execution commands for each machine and each parallel-programming model, as well as different job schedulers to submit and monitor parallel jobs.

All these issues outside the core course contents involve significant overhead time. These problems were overcome in 2003 at the University of A Coruña by applying grid computing technologies [10] to facilitate the use of the diverse parallel-pro-

cessing platforms. Specifically, a grid portal was developed that acts as a single point of entry to the geographically distributed computers used in the course and as a high-level, user-friendly environment for the students to manage transparently their parallel jobs. No reported practical experiences on the application of a grid portal to support the teaching of a course were found, but a few projects focused on setting up a grid infrastructure for teaching purposes, such as ISILab [11] and ULabGrid [12]. ISILab is a distributed environment that allows students to carry out remote laboratory experiments with real electronic instruments and circuits via the Web. The authors propose to map ISILab functionalities on a layered grid model: from the grid infrastructure composed of diverse resources geographically scattered (computers, storage systems, catalogues, networks, sensors, and devices) to the user application layer that controls the experiment execution. However, this grid-oriented approach is under development using proprietary protocols and application program interfaces (APIs), which limit reusability for other projects. ULabGrid is an architecture based on standard grid technologies that enables educators to design collaborative distant laboratories for students using a grid infrastructure. It allows students to run the software required for the laboratory from everywhere at any time. This approach was not conceived, however, to access supercomputing facilities, and it does not provide Web access via a portal. A prototype of this architecture applied to a flight simulator of an aeronautics laboratory is still being developed.

The remainder of the paper is organized as follows. Section II describes the contents of the course (both lectures and laboratory work) which provide the context for the design and development of the grid portal. Section III focuses on the portal: requirements, software architecture components, and functionalities provided to ease the completion of the course assignments. Finally, conclusions are drawn in Section IV.

II. Course Content

The one-semester course consists of 60 hours (40 hours in-class instruction and 20 hours laboratory). It attracts 20 students per year on average. All the students in the fourth year take a course in advanced computer architecture, which provides the architectural basis (including performance issues) for the parallel programming course. Thus, the students have a solid background in advanced topics, such as parallel computer architectures (shared memory and distributed memory), interconnection networks for parallel computers, cache coherence and memory consistency on multiprocessors, etc. Students are also skilled in C and Fortran 90 (taught in previous courses), which are the prerequisite languages for the course. Prior to the introduction of the parallel-programming course in 1998, the advanced computer architecture course included a short parallel-programming laboratory focused on the Parsys Supernode, a transputer-based machine (from 1990 to 1993), and PVM [13] message-passing programming on a network of workstations (1994–1997).

A. Course Lectures

The syllabus for the parallel-programming course currently being offered is on the following lines.

1) Introduction
   a) Course organization;
   b) Review of parallel computer architectures;
   c) Parallel-programming models;
   d) Case study: computing π in parallel using numerical integration.
2) Message-passing programming: MPI
   a) MPI main features;
   b) Point-to-point communications;
   c) Collective communications;
   d) Derived data types;
   e) Communicators and virtual topologies;
   f) Overview of new functionalities in MPI-2;

   Recommended textbook: [14]. Web material: [15].
3) Shared memory programming: OpenMP
   a) OpenMP main features;
   b) Directives to exploit loop-level parallelism;
   c) Directives for task-level parallelism;
   d) Synchronization mechanisms;
   e) Run-time library and environment variables;

   Recommended textbook: [16]. Web material: [17].
4) Data-parallel programming: HPF
   a) HPF main features;
   b) Data distribution and alignment directives;
   c) Directives and constructs to express data parallelism;
   d) HPF intrinsic and library procedures;
   e) Extrinsic procedures;

   Recommended textbook: [18]. Web material: [19].
5) Parallel-programming techniques: design of parallel programs
   a) Performance metrics for parallel programs;
   b) Data partitioning techniques;
   c) Techniques to enhance locality: exploiting memory hierarchy;
   d) Load balancing techniques;
   e) Case studies: parallel sorting, matrix multiplication, direct and iterative methods for solving linear systems of equations, parallel fast Fourier transform (FFT), and parallel tree search;

   Recommended textbook: [20]. Web material: [21].

The course ends with Lecture 5 because its contents may be too abstract for students at the beginning of the course without having experienced practical parallel-programming using the specific software taught in Lectures 2–4. Nevertheless, some topics of Lecture 5 are informally or intuitively introduced in the previous lectures, for instance, basic performance metrics such as speed-up or efficiency, load balancing techniques through OpenMP scheduling clauses, or data partitioning strategies using HPF directives. With this basis set, this lecture consolidates and completes the learning of parallel-programming techniques through practical case studies; thus, these topics are more easily assimilated by students.

TABLE I
SUPERCOMPUTING RESOURCES AVAILABLE FOR THE STUDENTS

Computer	Location	Programming	Short description
PC Network (sky.des.udc.es)	DES	MPI (MPICH)	10 PCs (PIII 800 MHz), Fast-Ethernet network
Self-made SCI cluster (muxia.des.udc.es)	DES	MPI (ScaMPI)	10 racked dual-processor nodes (PIV Xeon 1.8GHz), SCI network
Compaq Beowulf cluster (bw.cesga.es)	CESGA	MPI (MPICH-GM) HPF	16 racked single-processor nodes (PIII 1GHz), Myrinet 2000 network
SGI Origin 200 (silgar.cecafi.fi.udc.es)	SCS	OpenMP	ccNUMA multiprocessor (4 MIPS R10000 180 MHz)
HP Superdome (sd.cesga.es)	CESGA	OpenMP	cluster of 2 64-processor ccNUMA systems (Intel Itanium2 1.5 GHz)
Compaq HPC320 (sc.cesga.es)	CESGA	OpenMP/MPI hybrid	cluster of 8 4-processor SMP nodes (Alpha EV68 1 GHz)

B. Course Laboratory

Laboratory work consists of the following assignments.

1) *Training exercises*. The students are provided with the skeleton of several example codes of increasing difficulty to illustrate and experiment with the main concepts explained for each programming model.
2) *Comparison of programming models*. The students have to implement the same parallel algorithm (e.g., Gauss–Jordan elimination in the 2003 course) using MPI, OpenMP, and HPF in order to compare practically the programming features associated with each model.
3) *Algorithm performance contest*. A well-specified algorithm is proposed to be implemented in parallel (using MPI, since it provides more programming flexibility) on a particular supercomputer. The goal of the student is to maximize performance by taking into account both programming and architectural issues. Some examples used in previous years are parallel sorting algorithms, parallel matrix transposition, data redistributions (e.g., block-to-cyclic), or parallel prefix/suffix operations. This contest stimulates creativity, in-depth study, as well as experience sharing, discussion, and criticism (the best works are presented in class).
4) *Course miniproject*. It is an individual work proposed by the student under the supervision of the teacher who determines if it has sufficient technical merit. The goal of this miniproject is to put into practice as many theoretical contents as possible through the implementation of a parallel algorithm, application, or library using a particular programming model. Message-passing is usually the choice because students can install MPICH [15], a free implementation of MPI, on their own PCs for an initial development of their codes before moving them to a parallel computer. Students are encouraged to work on topics that have been studied in other courses (thus, these topics are familiar to them) to enhance interdisciplinary education. The most popular fields selected for the miniproject are numerical computation, neural networks, genetic algorithms, computer graphics, and image processing.

The resources currently available for the laboratory assignments are shown in Table I. The "Programming" column shows the parallel-programming software used for each computer, which does not indicate the only way to program that computer. Local resources are mainly devoted to training exercises and to the development and debugging of the other programming assignments before being tuned, executed, and evaluated on a particular CESGA supercomputer.

III. GRID PORTAL FOR THE COURSE LABORATORY

The laboratory resources listed in Table I (which may vary each year depending on availability) are heterogeneous and are administered by different institutions. These resources have been integrated into a grid environment for the parallel-programming laboratory. Grid computing [10] links different computing resources that belong to different institutions (usually sited in distant physical locations) and makes these resources available to users located at remote sites via public communication networks. Regarding this educational scope, the grid must provide the following basic functionalities.

- *Authentication*: Students and computing resources must be authenticated to allow the former to connect to the latter securely.
- *Information service*: An information service must provide students with both static and dynamic information about the setup and the status of the hardware and software resources of the grid (e.g., resource characteristics and availability).
- *Data transfer*: Students must be able to transfer securely and transparently their source codes and associated data files between the different computers of the grid.
- *Resource/job management*: A unified mechanism must provide for resource allocation, remote job submission, cancellation, monitoring, job input/output, etc.

These functionalities are provided by diverse software components that can be found integrated in software packages called grid toolkits. The Globus toolkit [22], which aims to use commodity software and technologies, is becoming the *de facto* standard for grid computing. It provides middleware services to

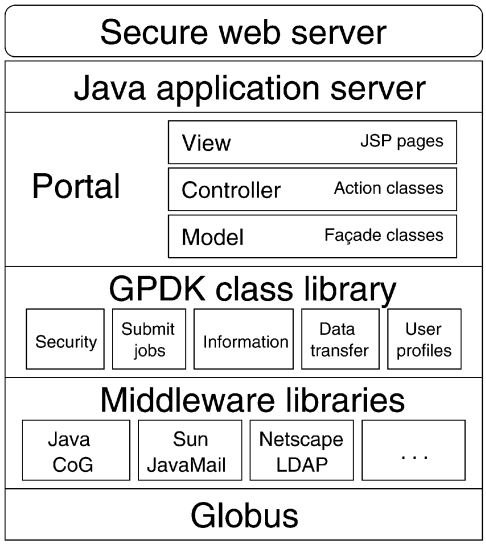

Fig. 1. Software architecture of the grid portal.

support grid environments. These toolkits are designed to provide basic grid services through low-level command-line interfaces, and thus a significant amount of learning time and effort must be spent by their potential users. This situation would be a serious obstacle to the effective adoption of the grid for instruction support, since the goal is precisely to enable students to develop more easily their assignments on the laboratory resources. To alleviate this problem, some easy and intuitive interfaces, such as grid portals that take the complexity of the grid away from the user, have been developed. A grid portal is a Web-based single point of entry to a grid and its implemented services. The Grid Portal Development Kit (GPDK) [23], a set of higher level function libraries that provide a generic framework for the development of grid portals, was used to implement the portal. The GridPort toolkit [24] also provides access to grid services through a Web interface by means of a set of Perl modules which wrap Globus. However, the design of GridPort makes code maintainability and portal customization more difficult than the approach taken by GPDK. A recent promising successor to GPDK is the GridSphere portal framework [25] with a new architecture based on portlets. The goal of this framework is to enable portal developers to implement new portal interfaces and add new functionality more quickly.

A. Portal Design and Architecture

A customized grid portal for laboratory support has been designed using several standard software components organized as a vertical stack, where each layer uses the services provided by the lower layer (Fig. 1). In the upper layer, the user opens a connection to the secure Apache Web server and makes an HTTP request. This request is forwarded to the Tomcat application server, which is running the portal Web application. Although GPDK provides a set of templates to generate ready-to-run portals, they have not been used since they implement an ad hoc architecture based on the use of program logic embedded in Java server pages (JSPs), which presents many drawbacks. For example, customization of the portal look for a particular site would probably involve changing some lines of Java code, which is generally outside of what a Website designer is willing to do. Conversely, adding new functionalities to the portal would require the portal developer to design its appearance at the same time. To avoid this issue, the Model-View-Controller (MVC) architecture [26] was used to provide a greater separation between business logic and presentation. The MVC architecture was used to design a new layer on top of GPDK (the Portal layer depicted in Fig. 1) and was implemented using the Java Struts framework. In the MVC architecture, the Model contains data and functionalities (e.g., file transfer, job submission…); the View is the part of the portal that interacts with the user (graphical display); and the Controller connects View and Model together and controls the execution of the portal. The Model was implemented using customized, high-level methods that make calls to functions of the lower layer, the GPDK class library that provides access to grid services along with some portal-specific utilities (e.g., portal user profiles). GPDK uses the libraries of the next layer: the Java Commodity Grid (CoG) kit [27] (which provides Java counterparts of the Globus C APIs) and some commodity libraries not specifically created for grid computing but used to access grid services [e.g., the Globus information service is based on the lightweight directory access protocol (LDAP)]. Finally, in the lowest layer lies Globus, which provides the basic grid services.

This layered approach allows the addition of new computing resources straightforwardly. When a new computer joins the grid, the associated information (e.g., number of processors or job managers installed) is automatically transferred from the Globus information service at the lowest layer of Fig. 1 to the upper layers, and the Portal View layer displays the interface to have access to the new resource. Moreover, the strong separation between layers provided by the MVC architecture allows the modification of the appearance of the portal (HTML code) without affecting the Java code that implements the functionalities and vice versa, and allows the inclusion of new functionalities easily.

B. Portal Features and Functionalities

The portal described in the previous subsection provides easy and transparent access to the available laboratory resources enumerated in Table I. The basic functionalities of the portal are depicted as a navigation diagram in Fig. 2. It is a UML (Universal Modeling Language) state diagram in which each portal page is represented by a state, and each link between pages is represented as a transition between states. The portal also includes links to useful course information: course schedule and news and course material (local and external) to enhance self-study.

The portal home page is shown in Fig. 3. Students must be authenticated and authorized to access the grid resources. The DES system administrator issues and stores credentials (a certificate and a private key with a lifetime of one semester) for each student on a credential server. There are two alternative ways for students to log into the portal: using a username and password (with a maximum lifetime of one week for security reasons), obtained by running the MyProxy package [28] on the credential server, or using a proxy credential file that must be copied to the computer where the student will launch the browser to access the portal (usually his/her home computer). This file, also with

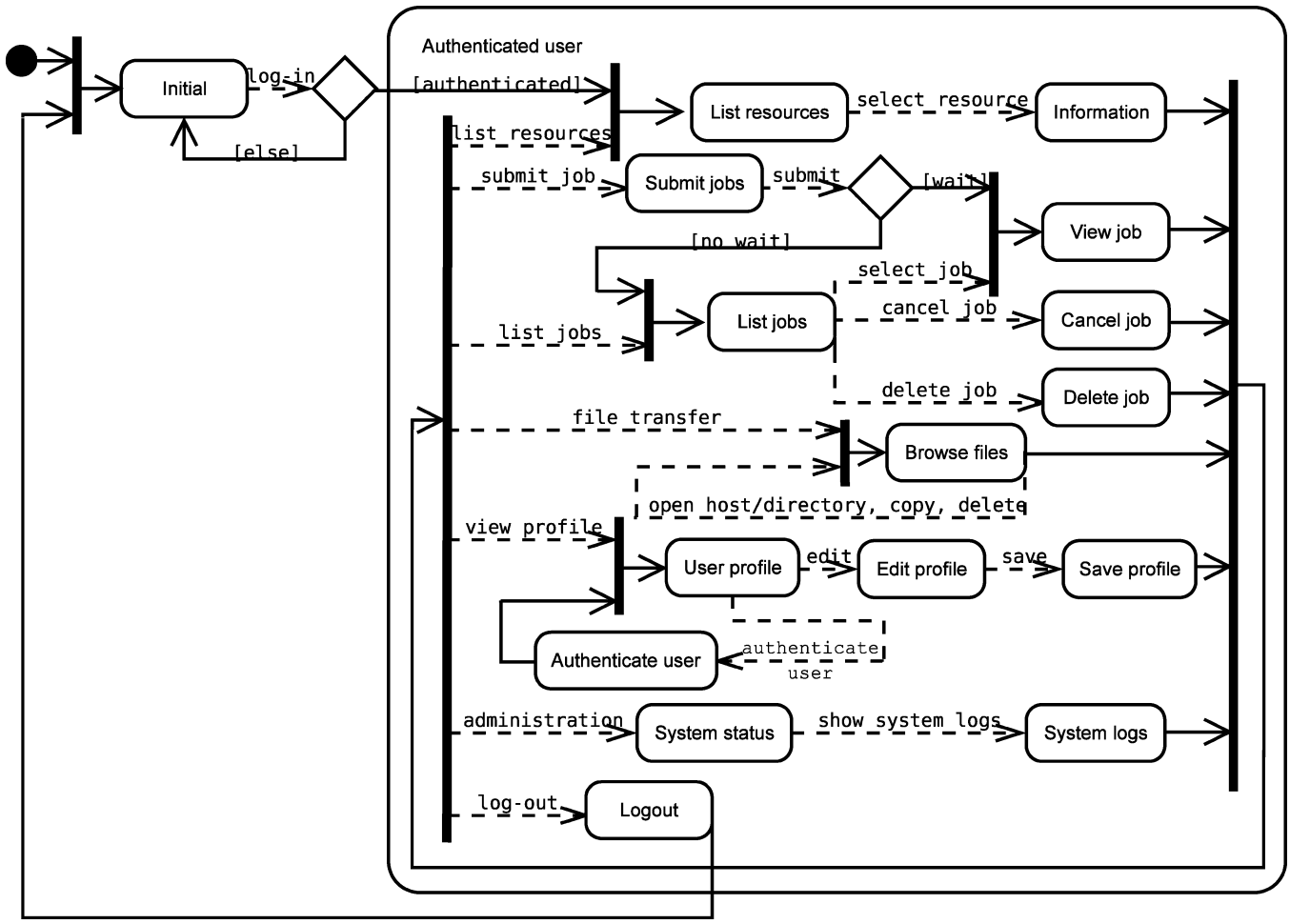

Fig. 2. Navigation diagram of the grid portal.

a limited lifetime, is obtained by running the grid-proxy-init Globus command on the credential server.

Once the student has been authenticated in the portal, the user's home page is displayed (see Fig. 4). This page shows the available resources (which could be different depending on the user) and detailed resource-level information, such as available processors and memory, average system load, or description of available job queues. This information is obtained dynamically from the Globus information service. The student has access to different functionalities from this page, such as file transfer between computers of the grid, job submission, or user profile information (personal data, certificate details, user preferences, or user history). For illustrative purposes, Fig. 5 shows the job submission page. It can be used to execute any program (operating system commands, compilation commands, parallel codes, etc.) on the selected machine of the grid. In this example, an eight-processor MPI job is submitted to the muxia cluster. This computer provides two different job managers (information obtained dynamically from the grid and displayed in a menu): fork, for running interactive jobs, and PBS (portable batch system), which was selected to launch the parallel job. The link "List jobs" in the menu displays a complete list of the user's jobs (see Fig. 6): job name, execution node, current status (active or completed), submission date, and a link to the job output (if available). There is also a link to cancel a running job or to delete an entry from the list when the job is finished.

In addition, the course teachers have access to several parameters about the portal activity ("System status" link): users currently connected, list of active and finished jobs, files transferred per user, login/logout records, and other activity logs. These parameters provide useful information on the individual and global use of the different computers of the grid. This information also turned out to be useful to detect and discourage incidents and bad practices in the use of supercomputers by students (e.g., massive submission of jobs, overallocation of processors and memory, or hung jobs as a result of careless job running).

Regarding the dynamics of the laboratory, students first develop and debug their assignments using local resources and submit their jobs to these resources in interactive mode through the portal. These jobs usually have small workloads and input data sets to check code correctness. Next, source codes are moved to CESGA supercomputers through the file transfer page. In the first contact with a real supercomputing environment, many students become impatient and complain because they are accustomed to obtain immediate results in the interactive execution on local resources. However, now their jobs are submitted to queues through the job submission page and, depending on the load of the supercomputer (which can be checked from the resource portal page), some time is required to gain resource access since CESGA supercomputers are used by researchers from the three Galician universities and from CSIC, a national research institute. Students are advised

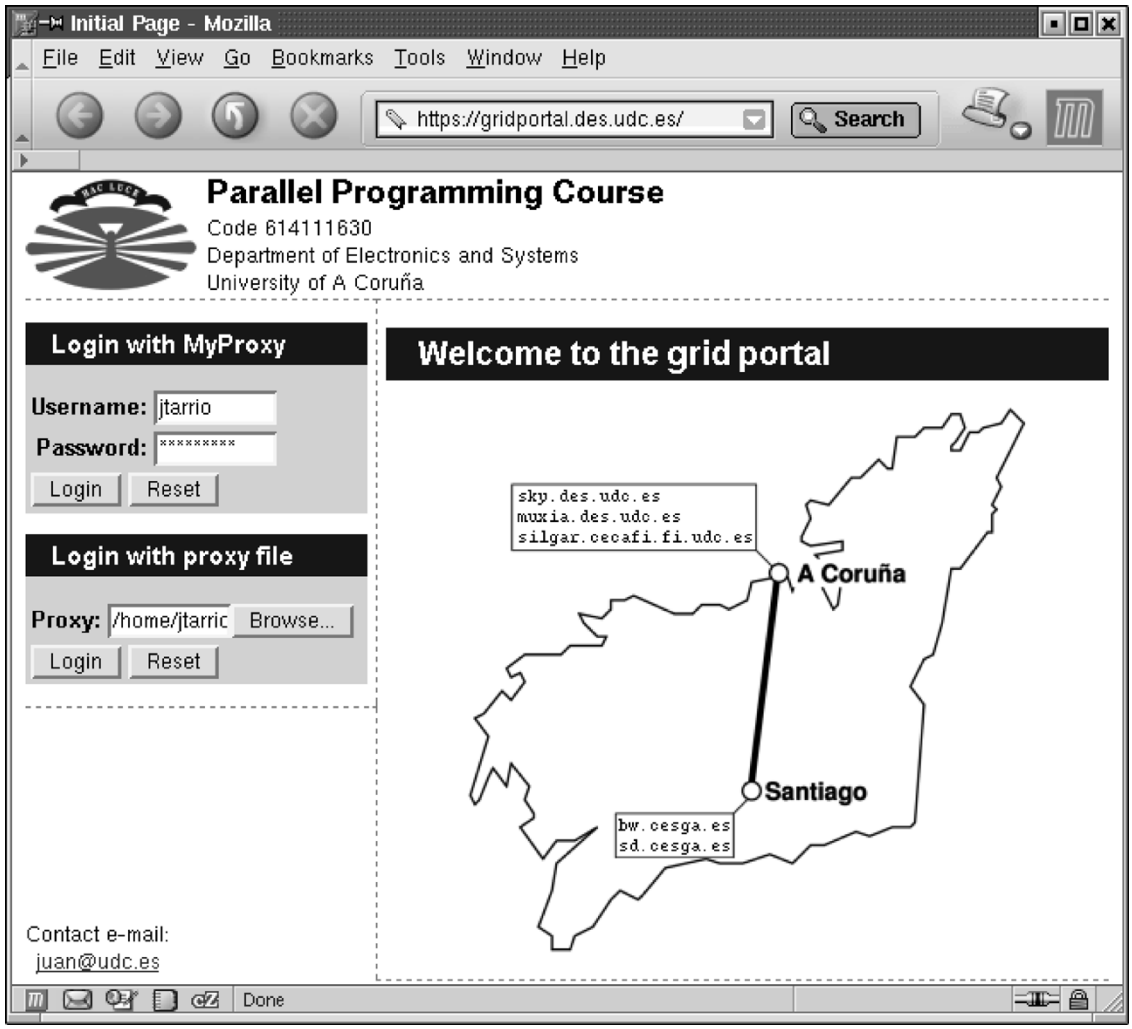

Fig. 3. Portal home page with authentication.

to finish their assignments far enough in advance to allow for supercomputers that may be overloaded close to the assignment deadline. Nonetheless, students rapidly adapt to this new environment and experience the power of supercomputers by using more processors and larger workloads for their jobs. They also use specific profilers (usually linked to their codes) that study the application behavior and report performance statistics (e.g., CPU time, communication time, or cache behavior) to guide performance tuning. At this stage, students become fully conscious that programming portability does not mean performance portability because performance results largely depend on the target machine. Students are provided in the lectures with basic information about available profiling tools or compiler options for each programming model, but they must experiment with them on their own. The effective use of these tools is an additional factor for the teacher to evaluate the assignments.

IV. Conclusion

In this paper, a grid portal to support an undergraduate parallel-programming course was described. Although grid technology is primarily intended to enable large-scale scientific research projects to better utilize and share distributed resources, it has been successfully applied in an educational environment. The portal provides an intuitive and homogeneous interface to the geographically distributed and dynamically changing supercomputing resources available for the laboratory assignments. Thus, the portal allowed students to focus on the course subjects instead of dealing with irrelevant and time-consuming technical issues, such as job submission or file transfer between computing resources, which helped to improve student performance.

The experience of the authors has shown that it is feasible for undergraduates to have access to supercomputing facilities and that their assignments do not interfere with the research activities of the supercomputing center. This situation results because of the relatively small number of students enrolled in the course, and in general, the assignments are designed so that they are not very CPU and memory consuming (a maximum of a few CPU hours, reasonable enough for students to exploit the power of the supercomputer). Moreover, the Supercomputing Center of Galicia (CESGA) enthusiastically supports the course as a medium to promote and extend the use of parallel computing technologies in Galicia, Spain.

The introduction of the portal was valued very highly by the students, who have even provided feedback on the portal design and functionalities. They highlighted the 24-hour access to supercomputing facilities from home, requiring only a

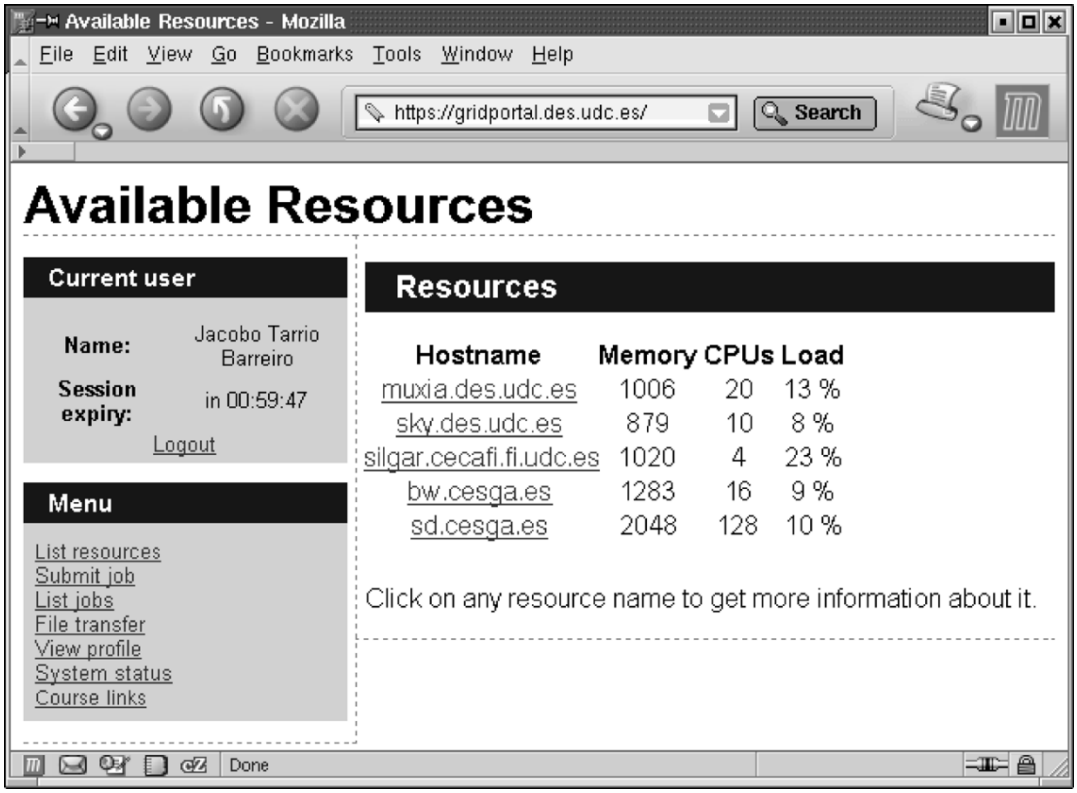

Fig. 4. User's home page: resource list and menu of functionalities.

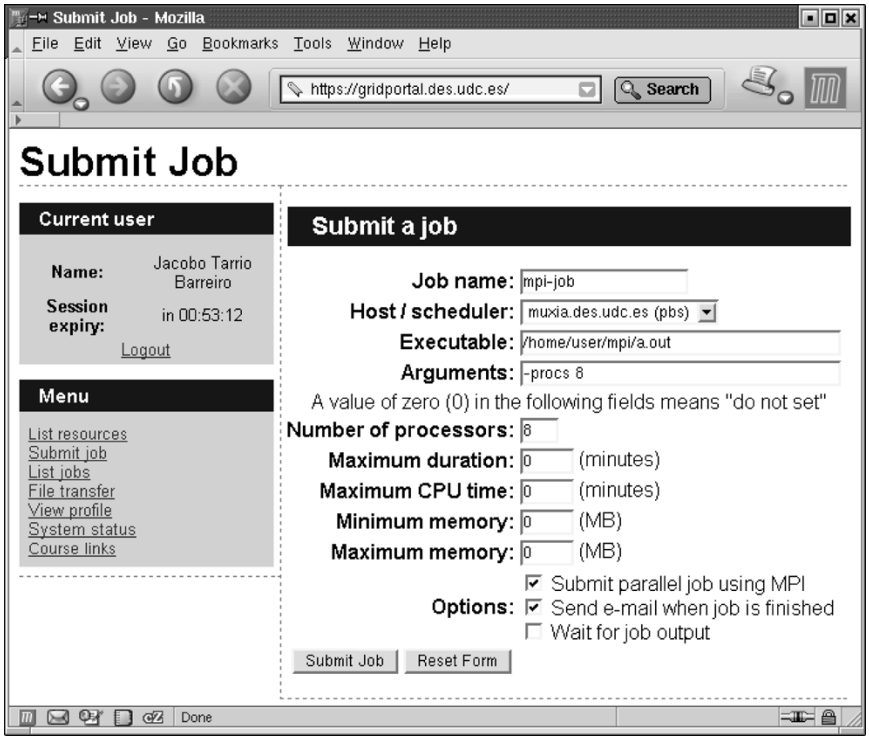

Fig. 5. MPI job submission through the portal.

browser, unlike other courses where hardware or software resources can only be accessed from on-campus machines. This fact, together with the motivation of programming production supercomputers, caused the students to spend many extra hours at home on the assignments, as revealed by the statistics provided by the portal. The format of the laboratory assignments described in Section II-B also stimulated self-study (mainly the course miniprojects) which, in some cases, became research. In

Fig. 6. List of user's jobs.

fact, this course turned out to be a good way to recruit Ph.D. students to work in the area of parallel processing.

Although the portal was developed for computer science (CS) undergraduates, it could be used at the graduate level not only for CS students but also for students from diverse disciplines of science and engineering (such as mathematics, physics, chemistry, biology, and civil, mechanical, or aerospace engineering), enrolled in a computational science/engineering program, who need parallel programming as an essential tool to solve large-scale problems in their fields. The portal also opens new possibilities for teaching parallel programming in the context of international Master's and Ph.D. programs as a means to enable universities from different countries share their supercomputing resources for teaching purposes. It could also provide a remote parallel-programming laboratory to students in countries that lack supercomputing infrastructures. The traditional educational links between Spain and Latin America offer a potential scenario for these purposes. Ongoing initiatives currently under development such as Gridcole [29], a collaborative learning environment that uses grid services technology, would be potentially appropriate in this context to provide integration of supercomputing capabilities or other specific resources at different locations in a wide area.

The development of the portal required a significant learning curve to acquire background in the technologies shown in Fig. 1. However, grid computing is still evolving, and it is expected that grid tools will be easier to set up and use so that they are more accessible to a wide audience not familiar with grid technologies but who might wish to build a grid portal for teaching purposes. Once built, a key issue is to obtain organizational support to manage the portal. Thus, currently, the departmental system administrator performs basic portal management tasks, such as portal accounting and authentication (e.g., student credentials), portal availability (maintenance of Web, application, and credential servers), or incident reporting on the grid infrastructure (e.g., unavailability of supercomputing resources). The URL for the portal home page is http://gridportal.des.udc.es

ACKNOWLEDGMENT

The authors would like to thank the Supercomputing Center of Galicia (CESGA) for providing access to its supercomputing facilities.The authors also would like to thank the Editor-in-Chief and the three anonymous referees for their helpful comments.

REFERENCES

[1] R. M. Butler, R. E. Eggen, and S. R. Wallace, "Introducing parallel processing at the undergraduate level," in *Proc. 19th ACM Tech. Symp. Computer Science Education*, Atlanta, GA, Feb. 1988, pp. 63–67.

[2] T. Hintz, "Introducing undergraduates to parallel processing," *IEEE Trans. Educ.*, vol. 36, no. 1, pp. 210–213, Feb. 1993.

[3] W. E. Toll, "Decision points in the introduction of parallel processing into the undergraduate curriculum," in *Proc. 26th ACM Tech. Symp. Computer Science Education*, Nashville, TN, Mar. 1995, pp. 136–140.

[4] C. H. Nevison, "Parallel computing in the undergraduate curriculum," *IEEE Computer*, vol. 28, no. 12, pp. 51–56, Dec. 1995.

[5] J. Adams, C. Nevison, and N. C. Schaller, "Parallel computing to start the millennium," in *Proc. 31st ACM Tech. Symp. Computer Science Education*, Austin, TX, Mar. 2000, pp. 65–69.

[6] (2004, Nov.). Supercomputing Center of Galicia. [Online]. Available: http://www.cesga.es

[7] (2004, Nov.). IEEE Task Force on Cluster Computing. [Online]. Available: http://www.ieeetfcc.org

[8] J. A. Youssefi and K. Zemoudeh, "A course in parallel processing," *IEEE Trans. Educ.*, vol. 40, no. 1, pp. 36–40, Feb. 1997.

[9] B. Wilkinson and M. Allen, "A state-wide senior parallel programming course," *IEEE Trans. Educ.*, vol. 42, no. 3, pp. 167–173, Aug. 1999.

[10] *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds., Morgan Kaufmann, San Francisco, CA, 1998.

[11] A. Bagnasco and A. M. Scapolla, "A grid of remote laboratory for teaching electronics," presented at the 2nd Int. LeGE-WG Workshop e-Learning and Grid Technologies, Paris, France, Mar. 2003.

[12] O. Ardaiz, P. Artigas, L. Díaz de Cerio, F. Freitag, A. Gallardo, R. Messeguer, L. Navarro, D. Royo, and K. Sanjeevan, "ULabGrid, An infrastructure to develop distant laboratories for undergrad students over a grid," in *Proc. 1st Eur. Across Grids Conf.*, vol. 2970, Lecture Notes in Computer Science, Santiago de Compostela, Spain, Feb. 2003, pp. 265–272.

[13] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing*. Cambridge, MA: MIT Press, 1994.

[14] P. S. Pacheco, *Parallel Programming with MPI*. San Francisco, CA: Morgan Kaufmann, 1997.

[15] The Message Passing Interface (MPI) Standard (2004, Nov.). [Online]. Available: http://www-unix.mcs.anl.gov/mpi

[16] R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, and R. Menon, *Parallel Programming in OpenMP*. San Francisco, CA: Morgan Kaufmann, 2001.

[17] (2004, Nov.). OpenMP Architecture Review Board. [Online]. Available: http://www.openmp.org

[18] C. H. Koelbel, D. B. Loveman, R. S. Schreiber, G. L. Steele, Jr., and M. E. Zosel, *The High Performance Fortran Handbook*. Cambridge, MA: MIT Press, 1994.

[19] The High Performance Fortran Home Page (2004, Nov.). [Online]. Available: http://dacnet.rice.edu/Depts/CRPC/HPFF

[20] B. Wilkinson and M. Allen, *Parallel Programming Techniques and Applications using Networked Workstations and Parallel Computers*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[21] I. Foster, *Designing and Building Parallel Programs*. Boston, MA: Addison-Wesley, 1995.

[22] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *Int. J. Supercomputer Applicat.*, vol. 11, no. 2, pp. 115–128, 1997.

[23] J. Novotny, "The grid portal development kit," *Concurrency Computation: Practice Experience*, vol. 14, no. 13–15, pp. 1129–1144, Nov./Dec. 2002.

[24] M. Thomas, S. Mock, M. Dahan, K. Mueller, D. Sutton, and J. R. Boisseau, "The GridPort toolkit: A system for building grid portals," in *Proc. 10th IEEE Symp. High Performance Distributed Computing*, San Francisco, CA, Aug. 2001, pp. 216–227.

[25] J. Novotny, M. Russell, and O. Wehrens, "Gridsphere: A portal framework for building collaborations," in *Proc. 1st Int. Workshop Middleware for Grid Computing, ACM/IFIP/USENIX Int. Middleware Conf.*, Rio de Janeiro, Brazil, Jun. 2003, pp. 178–185.

[26] I. Singh, B. Stearns, and M. Johnson, *Designing Enterprise Applications with the J2EE Platform*. Boston, MA: Addison-Wesley, 2002.

[27] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java commodity grid kit," *Concurrency Computation: Practice and Experience*, vol. 13, no. 8–9, pp. 645–662, Jul./Aug. 2001.

[28] J. Novotny, S. Tuecke, and V. Welch, "An online credential repository for the grid: MyProxy," in *Proc. 10th IEEE Int. Symp. High Performance Distributed Computing*, San Francisco, CA, Aug. 2001, pp. 104–111.

[29] M. L. Bote-Lorenzo, D. Hernández-Leo, Y. A. Dimitriadis, J. I. Asensio-Pérez, E. Gómez-Sánchez, G. Vega-Gorgojo, and L. M. Vaquero-González, "Toward reusability and tailorability in collaborative learning systems using IMS-LD and grid services," *Adv. Technol. Learning*, vol. 1, no. 3, pp. 129–138, 2004.

Juan Touriño (M'01) received the B.S., M.S., and Ph.D. degrees in computer science from the University of A Coruña, A Coruña, Galicia, Spain, in June 1993, October 1993, and 1998, respectively.

Since 1993, he has been on the faculty of the Department of Electronics and Systems at the University of A Coruña, where he is currently an Associate Professor of Computer Engineering, teaching undergraduate and graduate courses on parallel programming and advanced computer architecture. He also collaborates with Hewlett-Packard (HP) Spain on designing and teaching training courses for programming HP supercomputers. Since 1999, he has also been the Coordinator of his university with the Supercomputing Center of Galicia to advise researchers on the use of supercomputing facilities. He has been a Visiting Researcher at the Edinburgh Parallel Computing Center, Edinburgh, U.K., and at the CINECA Supercomputing Center, Bologna, Italy. His research interests include parallel algorithms and applications, grid computing, parallelizing compilers, performance evaluation of supercomputers, and management of large-scale parallel and distributed systems. He is coauthor of more than 60 technical papers on these topics.

Dr. Touriño is a Member of the IEEE Computer Society and the Association for Computing Machinery (ACM).

María J. Martín received the B.S., M.S., and Ph.D. degrees in physics from the University of Santiago de Compostela, Santiago de Compostela, Spain, in 1993, 1994, and 1999, respectively.

She is an Associate Professor of Computer Engineering at the University of A Coruña, A Coruña, Galicia, Spain, where she teaches undergraduate and graduate courses on computer architecture. She has been a Visiting Researcher at the Edinburgh Parallel Computing Center, Edinburgh, U.K., and at the Department of Computer Science of Stanford University, Stanford, CA. Her major research interests include parallel algorithms and applications, parallelizing compilers, grid computing, and fault tolerance for message-passing applications.

Jacobo Tarrío received the B.S. degree in computer science from the University of A Coruña, A Coruña, Galicia, Spain, in 2003.

He is currently an Open-Source Software Engineer in Alfa21, an information technology and network security company specializing in Unix/Linux systems. He has also been a Debian developer since 2001. He has been involved in the design and development of grid portals at the Department of Electronics and Systems of the University of A Coruña.

Manuel Arenaz received the B.S., M.S., and Ph.D. degrees in computer science from the University of A Coruña, A Coruña, Galicia, Spain, in 1997, 1998, and 2003, respectively.

He is an Assistant Professor of Computer Engineering with the University of A Coruña. His major research interests include optimizing and parallelizing compilers, parallel algorithms and applications, and grid computing.